Serial No. 09/890,536
Group Art Unit 2194
Docket No: 5-ART

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPEAL BRIEF - 37 C.F.R.. 1.192

U.S. Patent Application **09/890,536** entitled,

"METHOD AND SYSTEM FOR DYNAMIC CONFIGURATION OF ACTIVATORS IN A CLIENT-SERVER ENVIRONMENT"

Real Party in Interest: IONA Technologies, Inc.

Related Appeals and Interferences:

None.

Status of Claims:

- Claims 1-10 are pending.
- Claims 1-10 are rejected under 35 U.S.C. § 103(a) as being unpatentable over "The Common Object Request Broker: Architecture and Specification" (Corba).
- Claims 1-10 are hereby appealed.

Status of Amendments:

No after-final amendments were filed after the Final Office Action of 12/29/2006.

<u>SUMMARY OF THE CLAIMED SUBJECT MATTER</u>:

(**<u>NOTE</u>**: All references made in this section pertain to related **PCT/US00/02014** filed January 28, 2000)

According to claim 1, the present invention provides for a computer implemented method of activating a process, said method implemented via computer program code stored in said computer, said method comprising: generating one or more first plug-ins each configured to activate a target process **(see abstract, page 5, lines 1-5, lines 16-19 of PCT/US00/02014)**; dynamically registering the first plug-ins with a second plug-in **(see abstract, page 5, lines 1-5, lines 18-19, page 8, lines 21-22, page 16, lines 25-27, and page 19, lines 18-19 of PCT/US00/02014)**; permanently storing information relating to each registered first plug-in, and activating at least one target process based on said permanently stored information **(see abstract,**

page 5, lines 1-5, lines 16-19, page 8, lines 21-22, and page 16, lines 28-29 of PCT/US00/02014).

In addition to the features of claim 1, claim 2 further comprises the steps of: storing a flag for each registered first plug-in; perpetually activating the corresponding target process if the flag is set to a first state; and activating the corresponding target process upon a request if the flag is set to a second state **(see page 5, lines 6-9 and lines 20-23 of PCT/US00/02014)**.

In addition to the features of claim 2, claim 3, further comprises the steps of: generating an exception to indicate that a target process is inactive when its flag is not set to the first state or the second state **(see page 5, lines 10-13 and lines 24-26 of PCT/US00/02014)**.

In addition to the features of claim 1, claim 4 further comprises the steps of: providing a unique identifier for each target process; and sending and receiving a message between the first and second plug-ins using the identifiers **(see page 5, lines 10-13 and lines 27-29 of PCT/US00/02014)**.

In addition to the features of claim 4, claim 5 further comprises a method wherein the message includes information relating to a state change of the target processes, and wherein the state includes an activated state and a deactivated state **(see page 5, lines 29-30 of PCT/US00/02014)**.

According to claim 6, the present invention provides a server computer in a client-server computer system, comprising: a processing unit; and a storage device storing computer program code implementing at least: one or more first plug-ins each configured to activate a target process **(see abstract, page 5, lines 1-5, lines 16-19 of PCT/US00/02014)**, and a second plug-in configured to dynamically register the first plug-ins **(see abstract, page 5, lines 1-5, lines 18-19, page 8, lines 21-22, page 16, lines 25-27, and page 19, lines 18-19 of PCT/US00/02014)** and to permanently store information relating to the registered first plug-ins **(see abstract, page 5, lines 1-5, lines 16-19, page 8, lines 21-22, and page 16, lines 28-29 of PCT/US00/02014)**.

In addition to the features of claim 6, claim 7 teaches a server computer wherein the second plug-in comprises: a memory configured to store a flag for each registered first plug-in, wherein the second plug-in is further configured to perpetually activate target processes having their flags set at a first state and to activate target processes, upon receiving a request, having their flags set at a second state **(see page 5, lines 6-9 and lines 20-23 of PCT/US00/02014)**.

In addition to the features of claim 7, claim 8 teaches a server computer wherein the second plug-in is further configured to generate an exception to indicate that the target process is inactive when the flag is not set to the first state or the second state **(see page 5, lines 10-11 and 24-26 of PCT/US00/02014)**.

In addition to the features of claim 6, claim 9 teaches a server computer further comprising: a first computer program object configured to provide a unique identifier for each target process and configured to send a message using the identifiers **(see page 5, lines 10-13 and lines 27-29 of PCT/US00/02014)**.

In addition to the features of claim 9, claim 10 teaches a server computer wherein the message includes information relating a state change of the target processes, and wherein the state includes an activated state and a deactivated state **(see page 5, lines 29-30 of PCT/US00/02014)**.

<u>GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL</u>:

Claims 1-10 are hereby appealed.

- **<u>ISSUES</u>:**

Claims 1-10 are pending. Claims 1-10 are rejected under 35 U.S.C. § 103(a) as being unpatentable over "The Common Object Request Broker: Architecture and Specification" (Corba). <u>Was an improper 35 U.S.C. §103(a) rejection issued with respect to claims 1-10</u>?

- **<u>ARGUMENT</u>:**

<u>Was an improper 35 U.S.C. §103(a) rejection issued with respect to claims 1-10</u>?

Claims 1-10 are rejected under 35 U.S.C. § 103(a) as being unpatentable over "The Common Object Request Broker: Architecture and Specification" (Corba). To be properly rejected under 35 U.S.C. § 103(a), three basic criteria must be met. First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings. Second, there must be a reasonable expectation of success. Finally, the prior art reference (i.e., Corba) must teach or suggest all the claim limitations.

Additionally, the teaching or suggestion to make the claimed combination and the reasonable expectation of success must both be found in the prior art, and not based on Applicants' disclosure (In re Vaeck, 947 F.2d 488, 20 USPQ2d 1438 (Fed. Cir. 1991)). As per the arguments presented below, Applicants contend that the Examiner has failed to establish a prima facie case of obviousness under 35 U.S.C. §103(a).

The cited article "The Common Object Request Broker: Architecture and Specification" (hereafter, CORBA) outlines the CORBA standard and details how it defines APIs, communication protocol, and object/service information models to enable heterogeneous applications written in various languages running on various platforms to interoperate.

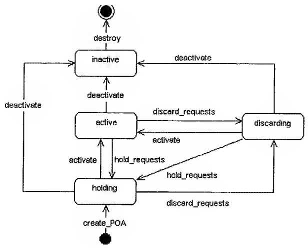
Applicants' independent claim 1, by contrast, provides for a computer implemented method of activating a process, said method implemented via computer program code stored in said computer, said method comprising: **generating one or more first plug-ins each configured to activate a target process**; **dynamically registering the first plug-ins with a second plug-in**; **permanently storing information relating to each registered first plug-in**, and **activating at least one target process based on said permanently stored information**.

With respect to the rejection of independent claim 1, on page 2 of the Final Office Action of 12/29/2006, the Examiner states that Applicants' feature of "generating one or more **first plug-ins**" can be equated to CORBA's created Portable Object Adapters (**POAs**) as described in section §9.2.3. On the same page of the Final Office Action, the Examiner cites §9.3.2 and §9.3.3 of CORBA as teaching the configuration of generated one or more plug-ins to activate a target process. For further support to his/her argument, the Examiner cites references to an "active state" described in §9.3.2 and §9.3.3 as teaching Applicants' feature of "configured to activate a target process". Applicants respectfully disagree with the Examiner's analysis as it is flawed.

As mentioned by the Examiner, §9.2.3 of CORBA does mention the creation of POAs.

However, it is respectfully noted that the Examiner's citations in §9.3.2 and §9.3.3 describe a "**POA Manager**" and **NOT**, as the Examiner asserts, a POA. The Board is respectfully requested to review §9.3.2 which, as the title suggests, outlines the **POA Manager Interface**. The Board is also respectfully requested to review §9.3.3 which, as the title suggests, outlines the **Adapter Activator Interface**. Further, the Board is respectfully requested to review Figure 9-3 titled "Processing States" which, by CORBA's own admission, outlines the **processing states associated with a POA Manager**. See also, for example, CORBA's own statement on page 9, which states that "A **POA Manager** has four possible processing states: active, inactive, holding, and discarding" (emphasis added). Hence, Applicants respectfully assert that while §9.2.3 of CORBA teaches the creation of POAs, and §9.3.2 and §9.3.3 deal with different entities, i.e., POA Manager Interface and Adapter Activator Interface, respectively. Therefore, **it would be erroneous to equate POA with the POA Manager Interface or the Adapter Activator Interface**.

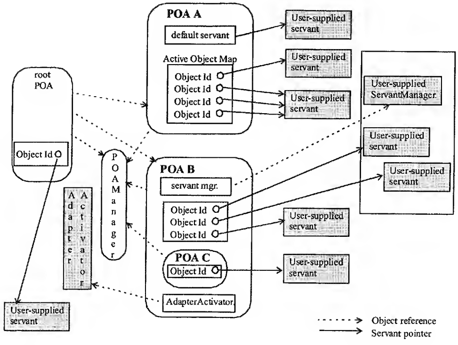
Further, the Examiner erroneously concludes that §9.3.2 teaches POAs that are configured to activate a target process based on the mere mention of the phrase "active state". However, the Board is respectfully requested to review figure 9-3 (reproduced below) and its associated description, which explicitly states that: (1) **these states are associated with the POA Manager**, and (2) **the "active state" reference used by the Examiner does NOT refer to activating a target process as the Examiner asserts, but merely refers to transitioning the POA Manager from a "discarding state" to an "active state"**.



<u>FIGURE 9-3 of CORBA</u>

Further, with respect to claim 1, the Examiner asserts that Applicants' feature of "dynamically registering the first plug-in with a second plug-in" is taught by CORBA's statement regarding "an AdapterActivator object must…registered with". The Examiner's citation appears in §9.3.3, merely mentions that "**<u>an AdapterActivator object must be local to the process containing the POA objects it is registered with</u>**". This merely mentions that the **<u>AdapterActivator object is registered with a *process* containing the POA objects</u>**. However, there is no teaching or suggestion for a **<u>first plug-in to dynamically register with a second plug-in</u>**.

The Examiner, in support of the rejection of claim 1, further asserts that CORBA in figure 9-2 and §9.2.3 teaches claim 1's feature of "**<u>permanently storing information related to each registered first plug-in</u>**". Applicants respectfully disagree with this assertion. Figure 9-2 (shown below) merely teaches Corba's architecture and §9.2.3 and makes no mention regarding **<u>permanently storing information related to each registered first plug-in</u>**.



<u>FIGURE 9-2 of CORBA</u>

Specifically, the Examiner's citation and the above-figure merely describe a "RETAIN policy" that relates to the maintenance of a map, labeled *ActiveObjectMap*, **<u>that associates or maps Object IDs with active servants</u>**. Such mapping between "Object IDs" and active servants cannot be equated to **<u>storing information related to each registered first plug-in</u>**.

Applicants agree with the Examiner that the CORBA reference fails to teach the permanent storage of information relating to each registered first plug-in. However, Applicants respectfully disagree with the Examiner that such a feature is remedied by the Cavanaugh reference. For support of his/her argument, the Examiner cites columns 7 and8 of Cavanaugh as teaching such permanent storage. However, by Cavanaugh's own admission, what is stored is "**<u>POA Name to POA ID mapping</u>**" (**<u>see column 7, lines 59-64 of Cavanaugh</u>**). Cavanaugh further gives examples of mappings such as "**Root/A/D -> 10**" and "**Root/B/D->11**". Given

Cavanaugh's own words that the **mapping between POA Name and POA ID** is what is stored, it would be erroneous to argue that Cavanaugh teaches or suggests **permanently storing information relating to each registered plug-in**. Similarly, it would also be erroneous to argue that Cavanaugh teaches **activating a target process based on the permanently stored information**.

Applicants, therefore, respectfully assert that the combination of CORBA and Cavanaugh cannot anticipate or render obvious Applicants' claim 1.

Applicants' independent claim 6, provides for a server in a client-server computer system, comprising: a processing unit; and a storage device storing computer program code implementing at least: **one or more first plug-ins each configured to activate a target process**, and **a second plug-in configured to dynamically register the first plug-ins and to permanently store information relating to the registered first plug-ins**.

The above-mentioned arguments substantially apply to independent claim 6.

Further, with respect to the feature of "**a second plug-in configured to dynamically register the first plug-ins and to permanently store information relating to the registered first plug-ins**", the Examiner supplies the same citation (as was provided for claim 1) from CORBA as teaching such permanent storage. However, the Examiner fails to provide an explanation regarding how CORBA's **mapping between Object IDs and active servants** reads on **a second plug-in that itself permanently stores information related to registered plug-**

ins.

At least for the reasons set forth above, Applicants, respectfully assert that the combination of CORBA and Cavanaugh cannot anticipate or render obvious Applicants' independent claim 6.

The above-mentioned arguments substantially apply to claims 2-5 and 7-10 as they inherit all the features of the claim from they depend from (i.e., claims 1 and 6, respectively). Hence, Applicants respectfully request the Examiner to withdraw the 35 U.S.C. §103 rejection with respect to claims 2-5 and 7-10, and hereby respectfully request allowance thereof.

In addition, with respect to dependent claim 2 and 7, the Examiner once again cites the processing states shown in Figure 9.3 as teaching "storing a flag for each registered plug-in" and "perpetually activating the corresponding target process if the flag is set to a first state and activating the corresponding target process upon a request if the flag is set to a second state". As was mentioned earlier, Figure 9-3 merely depicts various states which are **transition states associated with the POA Manager**. The "**active state**" reference used by the Examiner **does NOT** refer to activating a target process as the Examiner asserts, but merely refers to transitioning the POA Manager from a discarding state to a active state. Therefore, it would erroneous to argue that the CORBA reference teaches activating the corresponding target process if a flag is set to a first state and activating the corresponding target process upon a request if the flag is set to a second state.

As the features of claims 2 and 7 have not been taught or suggested by the CORBA reference, it would be moot to argue that the features of dependent claims 3 and 8 and taught by the CORBA reference.

Applicants, therefore, respectfully assert that the combination of CORBA and Cavanaugh cannot anticipate or render obvious Applicants' claim 2-3 and 7-8.

With respect to claims 4-5 and 9-10, the Examiner equates the mere mention of the "delivery of requests to the POA" (see §9.3.3 of CORBA) to Applicants' feature of sending a message using the identifiers. §9.3.3 of CORBA merely mentions the delivery of a request during initialization of a POA and provides no teaching or suggestion for **sending messages using identifiers**.

Applicants, therefore, respectfully assert that the combination of CORBA and Cavanaugh cannot anticipate or render obvious Applicants' claim 4-5 and 9-10.

As many of the features of pending claims 1-10 are not taught by the combination of CORBA and Cavanaugh, Applicants respectfully assert that an improper 35 U.S.C. §103(a) rejection was given by the Examiner. Applicants respectfully request the board to reverse the Examiner's rejection with respect to claims 1-10.

SUMMARY

Applicant contends that the Examiner, in the office action of 12/29/2006, has failed to provide a prima facie case of obviousness under U.S.C. §103 as there is no suggestion or motivation, either in the cited references, or in the knowledge generally available to one of ordinary skill in the art, to modify the references to provide for the various features of the Applicant's invention. As has been detailed above, none of the references, cited or applied, provide for the specific claimed details of Applicant's presently claimed invention, nor render them obvious. It is believed that this case is in condition for allowance and reconsideration thereof and early issuance is respectfully requested.

As this Appeal Brief has been timely filed within the set period of response, no petition for extension of time or associated fee is required. However, the Commissioner is hereby authorized to charge any deficiencies in the fees provided, to include an extension of time, to Deposit Account No. 50-4098.

Respectfully submitted by
Applicant's Representative,

/ramraj soundararjan/

Ramraj Soundararajan
Reg. No. 53,832

IP Authority, LLC
9435 Lorton Market Street #801
Lorton, VA 22079
(571) 642-0033

April 30, 2007

CLAIMS APPENDIX:

1. (Previously Presented) A computer implemented method of activating a process, said method implemented via computer program code stored in said computer, said method comprising:

generating one or more first plug-ins each configured to activate a target process;

dynamically registering the first plug-ins with a second plug-in;

permanently storing information relating to each registered first plug-in, and

activating at least one target process based on said permanently stored information.

2. (Original) The method of claim **1** further comprising:

storing a flag for each registered first plug-in;

perpetually activating the corresponding target process if the flag is set to a first state; and

activating the corresponding target process upon a request if the flag is set to a second state.

3. (Original) The method of claim **2** further comprising:

generating an exception to indicate that a target process is inactive when its flag is not set to the first state or the second state.

4. (Original) The method of claim **1** further comprising:

providing a unique identifier for each target process; and sending and receiving a message between the first and second plug-ins using the identifiers.

5. (Original) The method of claim **4** wherein the message includes information relating to a state change of the target processes, and wherein the state includes an activated state and a deactivated state.

6. (Previously Presented) A server computer in a client-server computer system, comprising:

a processing unit; and

a storage device storing computer program code implementing at least:

one or more first plug-ins each configured to activate a target process, and

a second plug-in configured to dynamically register the first plug-ins and to permanently store information relating to the registered first plug-ins.

7. (Original) The server of claim **6** wherein the second plug-in comprises:

a memory configured to store a flag for each registered first plug-in, wherein the second plug-in is further configured to perpetually activate target processes having their flags set at a first state and to activate target processes, upon receiving a request, having their flags set at a second state.

8. (Original) The server of claim **7** wherein the second plug-in is further configured to generate an exception to indicate that the target process is inactive when the flag is not set to the first state or the second state.

9. (Original) The server of claim **6** further comprising:

a first computer program object configured to provide a unique identifier for each target process and configured to send a message using the identifiers.

10. (Original) The server of claim **9** wherein the message includes information relating a state change of the target processes, and wherein the state includes an activated state and a deactivated state.

<u>EVIDENCE APPENDIX</u>:

None

<u>RELATED PROCEEDINGS APPENDIX</u>:

None